# COMSC-210 Lab 0
# Console Programs

**GOOD PROGRAMMING PRACTICES** *show / hide*

**ABOUT THIS ASSIGNMENT**
In this three-part lab assignment, you will develop simple "console programs". The purpose of this assignment is to learn the non-programming, logistical details for developing single-CPP projects, and submitting them for grading and credit. It is also your introduction to the basics of console I/O and other programming requirements that will apply to all future lab work in this COMSC course.

Refer to the "free programming resources" page on the Computer Science department's section of the DVC website, under "Programming how to's" for detailed steps in compiling C++ code in Windows, Mac OSX, or Linux/UNIX. It does not matter which compiler you use, because the CPPs that you develop are expected to work on any system with any C++ compiler.

This assignment is worth ZERO points. But it must be completed fully correctly before your first lab will be considered for scoring.

As you complete this assignment, post the required files to the COMSC server. You may post them all at once, or one-at-a-time as you complete them, as you prefer. The individual labs are graded in order, starting with "a", which must be fully correct before "b" is graded, and so on. This assignment is worth 0-points, to be awarded after all labs are successfully completed. Use the "Submit your FA2015 work" link on the class website to post your file for this lab, using the **lab0** folder provided.

**LAB 0a:** Who Are You? [ `AboutMe.txt` ]
Just so that you know how to submit your work, complete this non-programming exercise. With a code editor of your choosing, write a text file named **AboutMe.txt**. In this file, in any format you choose, include the following:

1. Your name
2. Prerequisites: When did you take COMSC 110 and 165? Who was your instructor? If you did not take COMSC 110 or 165, explain what equivalent course or preparation enabled you to fulfill the prerequisite requirement.
3. Compiler(s): What compiler(s) do you expect to use in this course?
4. Editor(s): What code editor(s) do you expect to use in this course?

*Post **AboutMe.txt** to the COMSC server.*

**LAB 0b:** The "Hello, World" Program [ `HelloWorld.cpp` ]
This is the place to start, when you learn any new computer language, or start using any new programming tool. It's meant to get the logistical details out of the way, so that the programmer can focus on programming logic and syntax.

Using C++ language syntax, write a program named **HelloWorld.cpp**. This is just about the simplest C++

program that there is. It prints in a "console" window the title of the assignment, "Lab 0b", and the programmer's name.

**UNIVERSAL REQUIREMENTS**
Observe these rules in all work that you submit for this course.

1. Spell and case this and ALL filenames in this COMSC course exactly as specified.
2. Note that we are using the `using namespace std;` statement. Do NOT use `using std::` statements.
3. Use 2-space indenting -- do NOT use TABs, unless you set your code editor to insert spaces for TABs.
4. Write your code in "code blocks". Include a // comment heading for each code block to explain what it does. Separate code blocks with single skipped lines, so that it is easy to see where one code block ends and the next begins.
5. Identify yourself and your program in TWO ways in each CPP you write. Do so once in // comments at the top of your CPP file, per the example in lab 0b below, and do so again in `cout` statements in your program.
6. Code::blocks is *not* a compiler. It's an editor. So if you use Code::blocks, you can include `// Editor used: Code::blocks`. But the compile Code::blocks uses is either a GNU or Microsoft compiler. Figure out which one your Code::blocks is using, and report it in `// Compiler used...`
7. Apply the code blocks from the Burns' Comsc-110 textbook, chapter 5 for all console and file input and output, modified to apply the "string buffer method" as explained in the lecture topic 0 notes, in all assignments that use `cin` to input numeric values.
8. Follow the **DO**'s and **DO NOT**'s given in each assignment, and apply them to all future assignments.

Type the following, with no indenting on the first line of coding. **Use 2-space indenting**. Do NOT use tabs in this course, except as you may have configured your code editor to insert spaces when you press the tab key. Skip single lines where indicated, for spacing. **Configure your code editor to insert SPACES instead of TABS, and to insert 2 of them.**

```
// Lab 0b, The "Hello, World" Program
// Programmer: YOUR NAME HERE
// Editor(s) used: XP Notepad
// Compiler(s) used: VC++ 2010 Express

#include <iostream>
using namespace std;

int main()
{
  // print my name and this assignment's title
  cout << "Lab 0b, The \"Hello, World\" Program \n";
  cout << "Programmer: YOUR NAME HERE\n";
  cout << "Editor(s) used: XP Notepad\n";
  cout << "Compiler(s) used: VC++ 2010 Express\n";
  cout << "File: " << __FILE__ << endl;
  cout << "Compiled: " << __DATE__ << " at " << __TIME__ << endl << endl;

  // a code block
 design and add a code block to read input from cin and echo it to cout...
}
```

**DO** use 2-space indenting.
**DO NOT** use tabs for indenting.
**DO NOT** use 4-spaces for indenting.
**DO NOT** replace __FILE__, __DATE__, or __TIME__ with the actual filename, date, or time.
**DO** copy/paste the "File:" and "Compiled:" lines exactly as they are, without modification.
**YOU MAY** put the cout's in a function and call that function from main, but if you do, you **must** write its prototype above main and its definition below main.
**DO NOT** leave more than one blank line to separate blocks of code.
**DO NOT** leave a blank line after an opening curly brace or before a closing curly brace.

**COMPILING**
Compile with the Microsoft Visual C++ `cl` or GNU `g++` command-line command, depending on the system you are using. Ask the instructor for a free g++ Windows compiler, that he can copy to your flash drive during any lab period in class. Or anytime you don't have a compiler at all, use ideone.com to compile and test your program. If compilation is not successful, make corrections and repeat until successful.

Compilation should create two files -- an "object file" (with the extention `.obj` with Visual C++, or `.o` in g++), and an "executable" (with the extention `.exe` in Windows). There is no need for saving object files. Try putting your executable on your desktop as an icon -- see what happens if you try to run it by double-clicking the icon. (Hint: the window will probably close as soon as it opens!)

If you prefer to use an IDE, you may -- but make sure that you understand how to use it and how it manages files. The lectures and examples in this course will all use command-line compiling, with various text editors. If you do use an IDE, make sure that you set its active configuration to "release mode" instead of "debug mode". If you must do step-by-step debugging, temporarily switch to "debug mode", but return to "release mode" when you are done debugging. Note that IDEs manage memory differently in "release mode" and "debug mode", and programs you develop in "debug mode" may not work the same way when you make a "release mode" version. If you discover a memory error at that time, it'll be too late to debug it easily.

**PROGRAMMING CONVENTIONS**
**stdafx.h and pragma**. Do NOT use either of these Microsoft-specific statements in ANY of programs for this COMSC course. If you are using an IDE, and it's adding them for you, and you can't stop it, then maybe you should not be using that IDE...

**iostream vs iostream.h**. Do NOT use the pre-standard C++ iostream.h in ANY of your work in this COMSC course. If you have a .h in a C or C++ library name, you are using the wrong version of the library.

**cmath vs math.h**. The C language libraries ending in ".h" have been replaced in standard C++. The new versions of these libraries drop the ".h" and prepend a "c". When you write a CPP (or H) file for this COMSC course, ALWAYS use these new versions of C-libraries -- that is, the ones starting with the letter "c" and not ending in ".h". Place the #includes for C-libraries AFTER the `using namespace std;`statement, with C++ libraries above.

**using namespace std;**. Use `using namespace std;` in all your work in this COMSC course.

**using std vs std::**. While it is possible to leave out `using std::cout;` and similar statements, and write (for example) `std::cout` instead of just `cout` wherever it appears in your code, we are NOT doing that in this COMSC course. Always use the `using namepace std;` statement.

**void main**. Do NOT use `void main()` because that is not standard C++.

**return 0**. Do NOT use `return 0;` as the last statement in `int main()` in ANY of your programs for this COMSC course. It's not required by the C++ language.

**endl vs \n**. You may print `\n` to do the same thing that `endl` does. The difference is that `endl` also "flushes the output buffer", while `\n` does not. It's okay to use `\n` in your work in this COMSC course, in place of `endl`. Make sure that the *last-executed* output statement in ALL your programs ends with `\n` or `endl`.

*Post **HelloWorld.cpp** to the COMSC server.*

---

**LAB 0c:** Console Programming Basics
Write a C++ console program named **TheBasics.cpp**, to read and print two sets of inputs, with two calculated values.
Write a C++ console program named **TheBasics.cpp**, to read and print two sets of inputs, with two calculated values. Refer to Burns' Comsc-110 textbook, chapter 5, for syntax -- linked to as I/O PDF in the lecture topic 0 notes. The first set is the user's age (in whole number years) and name (first and last), in that order, each on its own separate line, following its prompt. The second set is the current outside temperature (in floating point degrees F), and the user's location (city), in that order, each on its own separate line, following its prompt. For example:

```
Enter your age: 21
Enter your name: Joe Student
Enter the temperature outside right now (degrees F): 45
What city are you in right now? Walnut Creek
```

Once both sets have been read in, calculate two values: (1) the user's age one year from now, and (2) the temperature in degrees C. Finally, print the 4 input values and the two calculated values in the following 2-line format, spaced and punctuated per the example:

```
Joe Student is 21 years old now, and will be 22 a year from now.
It's 45 degrees F in Walnut Creek -- that's 7.2 degrees C.
```

Here are the specifications, in addition to the universal requirements and programming conventions explained in lab b above:

1. If the user enters a non-numeric for age or temperature, the program should not fail -- non-numeric entries should default to zero.
2. Allow for user name and city name to have more than one word in them, like Joe Student and Walnut Creek.
3. Print the degrees F in the same precision as entered (that is, if the input is 45.61, print it as 45.61).
4. Print the degrees C with one digit after the decimal point, rounded to the closest tenth of a degree.
5. Do NOT use the manipulator "fixed" -- use cout.setf instead, as explained in lecture.
6. Use ONLY int and double for numeric variables.
7. For text variables, use C strings, or C++ strings, or both, as you prefer.
8. Be sure to include your identifying information in both comment form and cout form.

Refer to your notes from lecture, that should explain how to do these.

**DO NOT** use stringstream. Use the "string buffer method" explained in the lecture notes.
**DO** refer to the "console and file I/O PDF" for how to do console input, and modify it for the "string buffer

method".
**DO NOT** use functions (except if you want to do so for your identifying cout statements).
**DO** test your program with at least the sample input sequence shown above.

HINTS: If you have to use if-statements in this one, you are not doing it right. Also, the use of "fixed" in `cout.setf(ios::fixed|ios::showpoint);` is NOT a manipulator -- you may use it. The "fixed" in `cout << fixed;` IS a manipulator -- do NOT use it.

*Post **TheBasics.cpp** to the COMSC server.*

**How to pause a console program:** *show / hide*

Click HERE for a YouTube-like video that demonstrates this. (Here's the URL for this link: cs.dvc.edu/youtube/Sharing.html.)

*hide*

**GRADING RUBRIC** *show/hide*